Beyond Entitlements for Cloud Native

Scalable Responsibility Management with Spring Boot and Open Policy Agent

Hong Liu and Chandra Guntur
Bank of New York Mellon
June 2019
Disclosure

BNY Mellon is the corporate brand of The Bank of New York Mellon Corporation and may be used as a generic term to reference the corporation as a whole and/or its various subsidiaries generally. Products and services may be provided under various brand names in various countries by duly authorized and regulated subsidiaries, affiliates, and joint ventures of The Bank of New York Mellon Corporation. Not all products and services are offered in all countries.

BNY Mellon will not be responsible for updating any information contained within this material and opinions and information contained herein are subject to change without notice.

BNY Mellon assumes no direct or consequential liability for any errors in or reliance upon this material. This material may not be reproduced or disseminated in any form without the express prior written permission of BNY Mellon.

©2019 The Bank of New York Mellon Corporation. All rights reserved.
About

Hong Liu

• Hong Liu is a Principal Developer in Resilient Systems Engineering, BNY Mellon.

• 18+ years of experience as a technologist using Java, with a recent focus on microservices and AI.

• Adept at creating plugins for IDEs such as Eclipse and IntelliJ IDEA.

• In her spare time, she likes to listen to classical music.

• Astronomy is her favorite theme to watch on TV.

Chandra Guntur

• Chandra Guntur is a Sr. Principal Architect and Java Advocate in Resilient Systems Engineering, BNY Mellon.

• Technologist in the financial services industry since 2003 and is programming with Java since 1998.

• One of the representatives for BNY Mellon in the Java Community Process (JCP) Executive Committee.

• JUG (Java User Group ) Leader, and helps run one of the largest Java user groups, NYJavaSIG (New York Java Special Interest Group).

• Frequent speaker at Java user groups, tech. conferences: Oracle CodeOne, Oracle Code NY, QCon New York, Devnexus and GIDS India.
Agenda

- Responsibility Management
- Technology Choices
  - HOCON, Open Policy Agent, Spring Boot, Eclipse Collections
- Architecture
- Code Samples
- OPA Policy Authoring Plugin for IntelliJ IDEA
Responsibility Management for the Enterprise

- A rationale
Why Responsibility Management – **Scenario 1**

- Service A needs to know if a user is a member of an enterprise **LDAP Group**
  - Access may be granted based on membership.
  - Access may be denied based on membership.
  - Access may be granted based on lack of membership.
  - Access may be denied based on lack of membership.
Why Responsibility Management – **Scenario 1**

- Service A needs to know if a user is a member of an enterprise **LDAP Group**
  - Access may be granted based on membership.
  - Access may be denied based on membership.
  - Access may be granted based on lack of membership.
  - Access may be denied based on lack of membership.

Then …

- Service B needs to know if a user is a member of an enterprise **LDAP Group**
Why Responsibility Management – Scenario 1

- Service A needs to know if a user is a member of an enterprise LDAP Group
  - Access may be granted based on membership.
  - Access may be denied based on membership.
  - Access may be granted based on lack of membership.
  - Access may be denied based on lack of membership.

Then …

- Service B needs to know if a user is a member of an enterprise LDAP Group

Questions

- How about Service/Application C, D or E?
- Who manages employees who move/leave/join the department/org/company (Movers/Leavers/Joiners)
Why Responsibility Management – Scenario 2

• Service M needs to know if a user is a member of an enterprise Email/AD Group

  • Access may be granted based on membership.
  • Access may be denied based on membership.
  • Access may be granted based on lack of membership.
  • Access may be denied based on lack of membership.
Why Responsibility Management – Scenario 2

• Service M needs to know if a user is a member of an enterprise Email/AD Group
  • Access may be granted based on membership.
  • Access may be denied based on membership.
  • Access may be granted based on lack of membership.
  • Access may be denied based on lack of membership.

Then …

• Service N needs to know if a user is a member of an enterprise Email/AD Group

Questions

• How about Service/Application O, P or Q?
• Who manages employees who move/leave/join the department/org/company (Movers/Leavers/Joiners)
Why Responsibility Management – **Scenario 3**

More complex evaluations occur as well.

Service X needs to check if all of the below are true for a user:
Why Responsibility Management – Scenario 3

More complex evaluations occur as well.

Service X needs to check if all of the below are true for a user:

- is member of LDAP Group 1
Why Responsibility Management – Scenario 3

More complex evaluations occur as well.

Service X needs to check if all of the below are true for a user:

- is member of LDAP Group 1
- is not member of LDAP Group 2
Why Responsibility Management – Scenario 3

More complex evaluations occur as well.

Service X needs to check if all of the below are true for a user:

- is member of LDAP Group 1
- is not member of LDAP Group 2
- is member of Email Group 1

≥ USD 200,000
Why Responsibility Management – **Scenario 3**

More complex evaluations occur as well.

Service X needs to check if all of the below are true for a user:

- is member of LDAP Group 1
- is **not** member of LDAP Group 2
- is member of Email Group 1
- is allowed to request an order of the amount USD 200,000
Why Responsibility Management – **Scenario 3**

More complex evaluations occur as well.

Service X needs to check if all of the below are true for a user:

- is member of LDAP Group 1
- is **not** member of LDAP Group 2
- is member of Email Group 1
- is allowed to request an order of the amount USD 200,000
- has at least two direct reports
Why Responsibility Management – Scenario 3

Questions

• What if each request is for different sets of groups and/or amounts?

• What if other services have similar functional constraints with different values?

• Where are such policies maintained, are they auditable and follow Config Management guidelines?

• Who manages Mover/Leaver/Joiner employees?

≥ USD 200,000
Why Responsibility Management – **Scenario 4**

Service Y needs to check responsibility privileges for a user/subject:
Why Responsibility Management – Scenario 4

Service Y needs to check responsibility privileges for a user/subject:

- in a given domain (Infra or Shared - service or tool)
Why Responsibility Management – **Scenario 4**

Service Y needs to check responsibility privileges for a user/subject:

- in a given **domain** (Infra or Shared - service or tool)
- for a given **cost code identifier** or org. business unit ($)

**Domain**

- organization
- environment
- action
- resource
Why Responsibility Management – Scenario 4

Service Y needs to check responsibility privileges for a user/subject:

- in a given **domain** (Infra or Shared - service or tool)
- for a given **cost code identifier** or org. business unit ($) 
- for a given **environment** (e.g. ‘PROD’, ‘QA’, ‘DEV’ …)
Why Responsibility Management – Scenario 4

Service Y needs to check responsibility privileges for a user/subject:

- in a given **domain** (Infra or Shared - service or tool)
- for a given **cost code identifier** or org. business unit ($)
- for a given **environment** (e.g. ‘PROD’, ‘QA’, ‘DEV’ …)
- for a given **action** (e.g. EDIT, DELETE, CREATE …)
Why Responsibility Management – Scenario 4

Service Y needs to check responsibility privileges for a user/subject:

- in a given **domain** (Infra or Shared - service or tool)
- for a given **cost code identifier** or org. business unit ($)
- for a given **environment** (e.g. ‘PROD’, ‘QA’, ‘DEV’ …)
- for a given **action** (e.g. EDIT, DELETE, CREATE …)
- for a given **resource** (e.g. org.databases.prod.instance1.schema1)
Why Responsibility Management – Scenario 4

Questions

• What if each request is for different sets of values for the given domain?

• What if other services have similar functional constraints with different values?

• Who manages Role-Responsibility per domain and User-Role Mappings?

• Who manages Mover/Leaver/Joiner employees?
Responsibility Management – **Common Solutions** – For Data

**DATA - External Services / Persistence**

- **LDAP/Active directory** queried by the application/service via **direct connections**.
- **User approver/manager** is queried via **proprietary corporate directory services**.
- **Role-Responsibility** mappings are usually stored in **local persistence** of the domain.
- **User-Role mappings** usually stored in any of: **local persistence, proprietary systems**.
Responsibility Management – **Common Solutions** – For Functions

**LOGIC - Calculations / Functions**

- Complex functions/calculations are **coded into the application/service**.
- Newer applications/services may separate such as an **independent microservice**.
- Some applications/services utilize **embedded rule engines** such as Drools.
- Some applications/services utilize **proprietary entitlement systems** for evaluations.
Responsibility Management Service

A solution to manage dynamic privileges and entitlements
Responsibility Management is performed via policies

Policies have a lifecycle

* More detailed flow in appendix
Responsibility Management System (RMS) – The Right Solution

A Responsibility Management System that:

- federates the calls to LDAP, Active Directory, and other services as integrated services
- provides appropriate mapping of roles and responsibilities, per domain
- provides for user to role mapping, per organization per domain
- provides proper SDLC and audit mechanism for policies per domain, to author and deploy
A Responsibility Management System that:

- provides for a **built-in policy engine** to evaluate complex calculations/functions using:
  - data provided as inputs by service-consumer
  - data queried from integrated services
  - policies provided by the domains
- caters to **applying a mover/leaver/joiner logic** to all controlled datasets
- provides **horizontal scaling** and thus, **high availability** for varying request volumes
BEFORE RMS

1. Entitlement System
   - URM
   - RRM

2. Roles System
   - RRM

3. Custom Service

4. DROOLS

- Application a
  - App Logic
  - LDAP Client
  - User Svc Client
  - URM via service, RRM via persistence

- Application b
  - App Logic
  - AD Client
  - User Svc Client
  - URM via persistence, RRM via service
  - Custom Service for policies

- Application m
  - App Logic
  - LDAP Client
  - AD Client
  - URM via persistence, RRM via persistence
  - Batch job to manage Users.

- Application n
  - App Logic
  - User Svc Client
  - URM via persistence, RRM via persistence
  - Drools rules for policies

- DB
  - URM
  - RRM

- LDAP

- AD

- User Svc

Decentralized Policies. Auditing is per-app. Bespoke User Mgmt.
WITH AN RMS

Centralized Policies.
Centralized Auditing.
Centralized User Mgmt.

RRM Role Responsibility Mapping
URM User Role Mapping
LDAP
AD
User Svc

Policy
Policy
Policy
Policy

Entitlement System
Roles System
DROOLS
Custom Service
Technologies Used

Technology choices for building the Responsibility Management Service
Payload format: HOCON format for payloads

A case for using Human-Optimized Configuration Object Notation

- Intent is to expose GET/POST operations.

- POST operations allow for a request body but do not support meaningful caching.

- Policy decisions should be queried (non-mutating), thus logically GET operations.

- GET operations do not support a request body.

- GET operations may be exposed to character limits, large parameter content not possible.

- JSON and individual query parameters are quite verbose.

- HOCON * trims the parameter verbosity by a significant amount.

https://github.com/lightbend/config/blob/master/HOCON.md
Payload format: HOCON benefits

Benefits of using Human-Optimized Configuration Object Notation

HOCON *

• syntax is quite simple and has low ambiguity.

• is a superset of JSON. JSON is parsed properly by HOCON parsers.

• allows the use of comments.

• allows multi-line strings.

• allows for includes and substitutions.

• has built-in durations (5d or 100ms)

https://github.com/lightbend/config/blob/master/HOCON.md
Payload format: HOCON features

Human-Optimized Configuration Object Notation – includes and substitutions

generic.conf

```{x: 10, y: ${x}, z: 5s}```

my.conf

```{a : { include "generic.conf" } }```
Payload format: HOCON compared to JSON

Sample comparisons

**Sample JSON**

```json
foo: {
  bar: {
    baz: myvalue
  }
}

employee: {
  firstname: "Jane"
  lastname: "Doe"
  nested: {
    loginTimeoutInMilliSeconds: 5000
  }
  fullname: "Jane Doe"
}

standard-policy: {
  developer: "yes"
  operator: false
}
```

**Sample HOCON**

```hocon
foo.bar.baz = myvalue

---- Or ----

foo {{ bar { baz = myvalue}}}

employee {
  firstname = "Jane"
  lastname = "Doe"
  nested {
    loginTimeoutInMilliSeconds = 5000
  }
  fullname = ${employee.firstname} ${employee.lastname}
}

standard-policy {
  developer = "yes"
  operator = false
}
```
Java Collections Library: Eclipse Collections

Key highlights

• Rich, concise and readable APIs.
• Clear mutable and immutable hierarchies for collection types.
• Memory efficient containers.
• Optimized eager APIs instead of Java Collection Framework’s lazy APIs.
• Improved code readability.
• Ease of learning thanks to several Code Katas.

https://www.eclipse.org/collections/
Policy Engine: Open Policy Agent (OPA)

Key highlights

• Open Policy Agent (OPA) * is an open source general purpose policy engine.
• Uses “rego” (inspired by Datalog) as a declarative native query language.
• Policies are written as rulesets (similar to functions).
• Policies can be queried as RESTful POST operations.
• Data and policy publishing is via RESTful PUT operations.
• Can be launched as a library for a service, an independent daemon or as a sidecar.
• Decision in RMS was to use OPA as a sidecar.

https://www.openpolicyagent.org/
OpenPolicyAgent usage

```
package domain1.policy1
import data.domain1.policy1.buckets
default allow = false
allow {
    buckets[i].name == input.bucket
    buckets[i].clients[j].name == input.client
    buckets[i].clients[j].access[k] == input.access
}
```

Query Payload

```
{ input {
    bucket: "bucket2",
    client: "client1",
    access: "READ"
}
}
```

Policy

```
[
  {
    "name": "bucket1",
    "clients": [
      {
        "name": "client1",
        "access": ["READ", "WRITE"]
      },
      {
        "name": "client2",
        "access": ["WRITE"]
      }
    ]
  },
  {
    "name": "bucket2",
    "clients": [
      {
        "name": "client1",
        "access": ["READ"]
      }
    ]
  }
]
```

data.json

```
http://localhost:8181/v1/data/domain1/policy1/allow
```

Policy Data Sidecar Query Payload data.json policy.rego
Architecting the Responsibility Management System

A platform solution for Responsibility Management
Responsibility Management System

Architecture (Version 1)

A Federated Responsibility Management Service
RMS Architecture – Version 1 (Federated)
RMS Architecture – Version 1 (Federated)
RMS Architecture – Version 1 (Federated)
RMS Architecture – Version 1 (Federated)

Policy Setup Process

Domain 1 Dev
- SCM
- Build Server
  - Domain 1 Policy 1
  - Domain x Policy 1
  - Domain 2 Policy 1

Rule Repository

Domain 2 Dev
- SCM
- Build Server

RMS Service

Responsibility Management

Policy Information Points (PIPs)

Open Policy Agent

Role Service
- RRM
- URM

User Service
- LDAP
- AD

Rule Repository

Pull
RMS Architecture – Version 1 (Federated)

RMS Service Consumers

Service 1

Service 2

Service x

Domain 1
Dev

SCM
Build Server

Domain 1
Policy 1

Domain 2
Dev

SCM
Build Server

Domain x
Policy 1

Domain 2
Policy 1

Policy Setup Process

Rule Repository

Policy Information Points (PIPs)

Responsibility Management

Open Policy Agent

RRM

URM

Role Service

User Service

LDAP

AD

Pull
Issues faced with a Federated Policy Management Architecture

Key issues

- Segregation and information-barrier needs implied more work.
- A rogue policy script could lead to loss of service for all domains.
- RM Service became the gatekeeper for testing and coverage.
- RM Service had to establish a release-train model to pick up new policies.
- Out-of-band policy changes lead to intermittent service-unavailability.
- Observation: Policy changes were more frequent when a new domain onboards.
Responsibility Management System

Architecture (Version 2)

A Distributed Responsibility Management Service
RMS Architecture – Version 2 (Distributed)

RMS Service Consumers

Service 1  Service 2  Service x

Policy Setup Process

Domain 1
Dev

SCM
Build
Server

Domain 2
Dev

SCM
Build
Server

... Domain x
Dev

SCM
Build
Server

Policy 1

Rule Repository

RMS Service

Policy
Information
Points (PIPs)

User
Service

LDAP

AD

Role
Service

RRM

URM

Service 1

Service 2

Service x

Service 1
RMS Architecture – Version 2 (Distributed)

RMS Service Consumers

Service 1
Service 2
Service x

Domain 1
Dev
SCM
Build Server

Domain 2
Dev
SCM
Build Server

... Domain x
Dev
SCM
Build Server

... Domain 2
Dev
SCM
Build Server

Policy Setup Process

Domain 1
Policy 1

Domain x
Policy 1

Domain 2
Policy 1

Rule Repository

Policy Administration Service (PAS)

RMS Service

Policy Information Points (PIPs)

User Service
LDAP
AD
Role Service
RRM
URM

Service 1
Service 2
Service x

Information Classification: Public
RMS Architecture – Version 2 (Distributed)

RMS Service Consumers

Service 1  Service 2  Service x

...  

Policy Setup Process

Domain 1
Dev

SCM
Build
Server

Domain 2
Dev

SCM
Build
Server

Domain 1
Policy 1

Domain x
Policy 1

Domain 2
Policy 1

Rule
Repository

RMS Service

Policy
Information
Points (PIPs)

User
Service

LDAP

AD

Policy
Administration
Service (PAS)

Role/Resp. (RR), User/Role (UR)

Mappings

Role/Resp., User/Role Mappings

Role Service

RRM

URM

Service 1

Service 2

Service x
RMS Architecture – Version 2 (Distributed)

RMS Service Consumers

Service 1
Service 2
Service x

... 

Policy Setup Process

Domain 1
Dev

Domain 2
Dev

SCM
Build
Server

SCM
Build
Server

Domain 1
Policy 1

Domain x
Policy 1

Domain 2
Policy 1

Rule
Repository

Publish Policy

Role/Resp., User/Role Mappings

Policy Administration Service (PAS)

Role/Resp. (RR), User/Role (UR)

Mappings

Policy Information Points (PIPs)

User Service

LDAP

AD

RMS Service
RMS Architecture – Version 2 (Distributed)

RMS Service Consumers

- Service 1
- Service 2
- Service x

... Policy Setup Process ...

- Domain 1
  - Dev
  - SCM
  - Build Server
  - Domain 1 Policy 1
  - Domain x Policy 1
  - Domain 2 Policy 1

- Domain 2
  - Dev
  - SCM
  - Build Server

RMS Service

- Policy Information Points (PIPs)
- LDAP
- AD

- Policy Administration Service (PAS)
- Policy Bundles Repository
- Policy Bundles
- Policy + RR & UR Mappings
- Role/Resp. (RR), User/Role (UR)
- Mappings
- Role Service
- RRM
- URM

- Publish Policy
- Role/Resp., User/Role Mappings
RMS Architecture – Version 2 (Distributed)

RMS Service Consumers

Service 1  Service 2  Service x

Policy Setup Process

Domain 1 Dev

- SCM
- Build Server

- Domain 1 Policy 1
- Domain x Policy 1
- Domain 2 Policy 1

Publish Policy

Rule Repository

Domain 2 Dev

- SCM
- Build Server

Policy Setup Process

Role/Resp., User/Role Mappings

Policy Setup Process

Policy Administration Service (PAS)

Policy + RR & UR Mappings

Role Resp. (RR), User/Role (UR) Mappings

Policy Distribution Service (PDS)

Policy Bundles

Policy Information Points (PIPs)

RMS Service
RMS Architecture – Version 2 (Distributed)

RMS Service Consumers

Domain 1
- Dev
- SCM
- Build Server
- Policy 1

Domain 2
- Dev
- SCM
- Build Server
- Policy 1

Domain x
- Policy 1

Service 1
- Service 2
- Service x

Policy Setup Process

Publish Policy

Rule Repository

Policy Administration Service (PAS)

Policy Distribution Service (PDS)

Policy Information Points (PIPs)

AD

LDAP

Policy Bundles Repository

Policy Bundles

Role/Resp. (RR), User/Role (UR)

Role/Resp., User/Role Mappings

Role Service

RRM

URM

Service 1

Service x
## Benefits of a Distributed Policy Management Architecture

Comparing Version 1 (federated single policy engine) with Version 2 (distributed policy engines)

<table>
<thead>
<tr>
<th>Feature</th>
<th>V1 Federated Policy Engine</th>
<th>V2 Distributed Policy Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segregation and Information Barriers</td>
<td>Requires <strong>additional work</strong></td>
<td>Is implicit, <strong>no additional work</strong></td>
</tr>
<tr>
<td>Impact of a rogue policy script</td>
<td>Outage for <strong>all domains</strong></td>
<td>Outage <strong>only for the specific domain</strong></td>
</tr>
<tr>
<td>Gatekeeping for testing and coverage</td>
<td>Requires <strong>RMS</strong> to be the gatekeeper</td>
<td>Requires <strong>domain</strong> to be the gatekeeper</td>
</tr>
<tr>
<td>Strategy for new and updated policies</td>
<td>Needed a <strong>Release Train model</strong></td>
<td>A domain can push policies on-demand</td>
</tr>
<tr>
<td>Impact of ad-hoc policy changes</td>
<td>RMS Downtime for <strong>all domains</strong></td>
<td>RMS Downtime for <strong>the changed domain</strong></td>
</tr>
<tr>
<td>Implicit RBAC Support</td>
<td><strong>-</strong></td>
<td><strong>Available</strong></td>
</tr>
</tbody>
</table>
Policy Bundles Repository

Policy bundles repository stored enriched policy archives.

Enriched policy bundles are archives that contain:

• Policy file(s), **specific to the domain**.

• Policy static data, **specific to the domain**.

• Standard RMS OPA policy rego files **common across all domains**.
Policy Bundles Repository

Folder structure in policy bundles repository:

- `<domain>`
  - `<policy>`
    - `<version>`
      - `<policy bundles>`

Example:

- `domain1`
  - `policy1`
    - `1.0.0`
      - `enriched-opa-bundle.tar.gz`
How the Policy Agent is setup

• Open Policy Agent (the executable)

• Open Policy Agent – Configuration

• Open Policy Agent – Dockerfile command
How the Policy Agent is setup – Configuration files

OPA Configuration file (located at \${configPath})

```
services:
  - name: domainPolicies
    url: policyDistributionServiceUrl/
    allow_insecure_tls: true
    bundle:
      name: policyDomain/policyName/policyVersion
      service: domainPolicies
      polling:
        min_delay_seconds: minDelaySeconds
        max_delay_seconds: maxDelaySeconds
```

Environment Variables
How the Policy Agent is setup – Dockerfile command

OPA launch command (used in the Dockerfile)

```
exec ./opa run --server --log-level=debug -c ${configPath}
```
RBAC Policy Library

```rego
package rbac

user_has_responsibility(userId, action, resource) {
    role := roles[_]
    responsibility := role.responsibilities[_]
    does_resource_match(resource, responsibility)
    responsibility.actions[_] = action
}

is_user_a_member(userId, role)

is_user_a_member(userId, role)

...  
```

Sample Role Data Excerpts

```json
{
    "name": "App User",
    "responsibilities": [
        {
            "resource": "service.1",
            "actions": ["provision"
        ]
    ],
    "members": ["EVERYONE"]
}
{
    "name": "App Admin",
    "responsibilities": [
        {
            "resource": "regexp:service\..*",
            "actions": ["create", "update", "delete", "view"
        ]
    ],
    "members": ["org:abc"]
}
```

Application Policy

```rego
package application1

default allow = false

allow {
    data.rbac.user_has_responsibility(
        input.userid, input.action, input.service)
}
```
OPA IntelliJ Plugin

A development tool for OPA language
OPA IntelliJ Plugin

- OPA IntelliJ Plugin is functional work-in-progress policy editor.
- The editor parses and validates OPA policy.
- Relies on the OPA language reference linked * below.
- Can be customized for editor color schemes in IntelliJ.
- Work continues on indentation, run configurations and variable tracking.

https://www.openpolicyagent.org/docs/latest/language-reference/
Before & After
OPA language validation
OPA language validation
OPA editor plugin color scheme

Select

- Preferences
  - Editor
    - Color Scheme
      - Open Policy Agent
In Summary

• Responsibility Management as a Service can resolve issues on several fronts.

• Choice of a payload format (HOCON over JSON or XML) can help control verbosity.

• Choice of architecture (federated versus distributed) can help determine resilience.

• Distributed policy engines can alleviate back-pressure and volume demands.

• Distributed policy engines can reduce outages and maintenance-related downtimes.

• Creating a policy editor plugin can help boost productivity.
Links

- HOCON
  
  https://github.com/lightbend/config/blob/master/HOCON.md

- Eclipse Collections

  https://www.eclipse.org/collections/

- Open Policy Agent

  https://www.openpolicyagent.org/
Appendix: Understanding Responsibility Management

Policy Administration Point
- Policy Authoring
- Policy Storage
- Policy Audit/Report

Policy Distribution Point
- Policy Bundling
- Policy Distribution

Policy Evaluation Point
- Policy Procurement
- Policy Evaluation

Policy Enforcement Point
- Policy Invocation
- Policy Application
- Policy Dynamic Inputs

Policy Information Point
- Policy Reference Data
- Policy Entitlements
- Policy Identities

Access Reconciliation Review & Certification
- Entitlements Discovery
- Access Reconciliation
- Access Certification

Enterprise Roles and Responsibilities
- Privileged Business Functions

Managed Provisioning
- Workflows
- Downstream Fulfilment